Task Planning Using Physics-based Heuristics on Manipulation Actions

Aliakbar Akbari, Muhayyuddin and Jan Rosell
Institute of Industrial and Control Engineering (IOC)
Universitat Politècnica de Catalunya (UPC) – Barcelona Tech
Barcelona, Spain, jan.rosell@upc.edu

***Abstract*—In order to solve mobile manipulation problems, the efficient combination of task and motion planning is usually required. Moreover, the incorporation of physics-based information has recently been taken into account in order to plan the tasks in a more realistic way. In the present paper, a task and motion planning framework is proposed based on a modified version of the Fast-Forward task planner that is guided by physics-based knowledge. The proposal uses manipulation knowledge for reasoning on symbolic literals (both in offline and online modes) taking into account geometric information in order to evaluate the applicability as well as feasibility of actions while evaluating the heuristic cost. It results in an efficient search of the state space and in the obtention of low-cost physically-feasible plans. The proposal has been implemented and is illustrated with a manipulation problem consisting of a mobile robot and some fixed and manipulatable objects.**

***Index Terms*—Task and motion planning, Manipulation, Physics-based heuristics.**

I. INTRODUCTION

Mobile manipulation planning requires, at task level, the finding of a sequence of actions, and at motion level, the finding of the way to execute them. Therefore, the efficient combination of both planning levels has currently emerged as a substantial challenge. In this line, some approaches like [1], [2], [3], [4] considered external modules, layers or mechanisms to interface between the symbolic and the geometric reasoning processes. Other approaches look for hierarchical planning solutions based on hierarchical planning, like [5], [6], [7], [8], that evaluate task-level decisions with low-level geometric-reasoning modules. Finally, some other approaches like [9], [10], [11] propose different ways to integrate geometric information within the Fast-Forward (FF) task planner [12], which has been recognized as one of the best symbolic task planners in Artificial Intelligence. The FF method is an heuristic-based symbolic task planner based on two main components. One is responsible for finding, for a given state being analyzed and using a relaxed version of the GraphPlan algorithm [13], an heuristic value of the cost to reach the goal, as well as the set of helpful actions, i.e., those actions that executed from that state have a high probability of being in the solution plan. The other is devoted to the search for the more promising successor state using the heuristic values and an Enforced Hill-Climbing method.

This work was partially supported by the Spanish Government through the projects DPI2013-40882-P and DPI2014-57757-R. Aliakbar Akbari is supported by the Spanish Government through the grant FPI 2015. Muhayyuddin is supported by the Generalitat de Catalunya through the grant FI-DGR 2014.

Complementarily, on the one hand, the ontology-based knowledge representation has been employed to enhance manipulation planning by easing the way to hand over useful information upon the planning phase (for instance, some approaches like [14], [15] tackle manipulation tasks involving housework activities by describing the robot manipulation world in terms of ontologies). On the other hand, the incorporation of a physics-based engine at the motion planning level has been proposed to enable and take into account the interaction between rigid bodies, in order to determine in a realistic way how to execute the symbolic actions obtained by the task planning level. For instance, the work in [16] used dynamic simulation to solve the problem of navigating among moveable obstacles that can be pushed, and the work in [17] proposed a variant of the Rapidly-exploring Random Trees (RRT, [18]) to consider interactions between rigid bodies, besides the kinodynamic (geometric and differential) constraints of the robot. Similarly, the approach in [19] also used a sampling-based planner with a dynamic engine as a state propagator. This planner was called *smart motion planner* because it incorporated ontological knowledge and a Prolog-based reasoning on how to interact with objects, in order to efficiently guide the exploration.

This combination of ontological knowledge-based task planning and physics-based motion planning was further explored in [20] and [21]. The former approach first determined a number of potential plans and then computed the accumulated cost of each plan execution, in order to identify the most feasible plan, by calling the low-level physics-based motion planer. The latter presented a simultaneous task and motion planning approach that used a physics-based motion planner to evaluate the feasibility of selected actions explored when planning at task level using GraphPlan, and to cut off the branches of those infeasible actions, thus making the exploration of the planning space more efficient. Both approaches use the physics-based motion planner to evaluate the cost of actions, and this makes them computationally costly. In order to mitigate this drawback, the present paper proposes a planning method based on the Fast-Forward, that only calls the motion planner for evaluating the selected actions found in the relaxed plan used to compute the heuristic cost to reach the goal from a given state being explored, without the need of calling the motion planner for any other action.

The paper presents a new framework for the efficient combination of knowledge-based task planning and physics-

based motion planning based on the FF planner. A reasoning process is performed on symbolic literals in terms of knowledge and geometric information about the workspace (offline reasoning), as well as using high-level reasoning and physics-based motion planning to determine the applicability of actions and to determine the feasibility of applicable actions along their effects (online reasoning). As a consequence the proposed method is able to prevent or discard some unnecessary actions while planning. Moreover the computation of the heuristic cost that guides the search of the solution plan takes into account the physical properties of objects and the actual cost of selected actions computed by a physics-based motion planner. The proposal, therefore, aims to make the planning more efficient and to find physically feasible low-cost plans.

II. Problem Formulation and Solution Overview

A. Scope and a motivating example

This paper deals with manipulation problems where a mobile robot is required to move from an initial region towards a goal one in an indoor environment cluttered with obstacles that can be either fixed or manipulatable. The existence of manipulatable obstacles may partition the free subspace of the configuration space of the robot, C_{free}, into disconnected regions C_i. If the initial and the goal regions of the query to be solved lie in different disconnected regions, then the robot will need to move some manipulatable obstacles away in order to connect them and find a solution path. It is assumed that the robot is able to perform three type of actions: *push* and *pull* actions to change the position of manipulatable objects, and the *transit* action to move freely along a collision-free path. It is also assumed that there are two types of obstacles in the environment, *fixed obstacles* which the robot must avoid, and *manipulatable obstacles* (*MObs*) that can be pushed or pulled by the robot along a given direction. In order to interact with a *MObs*, the robot must be located in the corresponding manipulatable region *MRgn*.

As an example, consider the task shown in Fig. 1 in which the robot is located in the initial region and must move towards the goal one. The initial and goal configurations of the robot do not belong to the same connected region of C_{free} and therefore some manipulatable objects, labeled from A to H, must be moved away. These objects must be manipulated through a handle that determines the motion direction, as well as the manipulation region (highlighted in dark red) where the robot must be located in order to pull or push them. Moreover, these objects have different physical features and some of them, like object A, may be beyond the robot manipulation capacity. Also, it must be noted that there can be some actions which do not provide fruitful effects to solve the problem. For example, pulling object F does not provide the access from C_2 towards C_3. Finally, note that a number of potential possible plans may exist and the least-cost one is the one sought.

These aforementioned issues pose interesting challenges that can be properly solved by considering an efficient combination of task and physics-based motion planning.

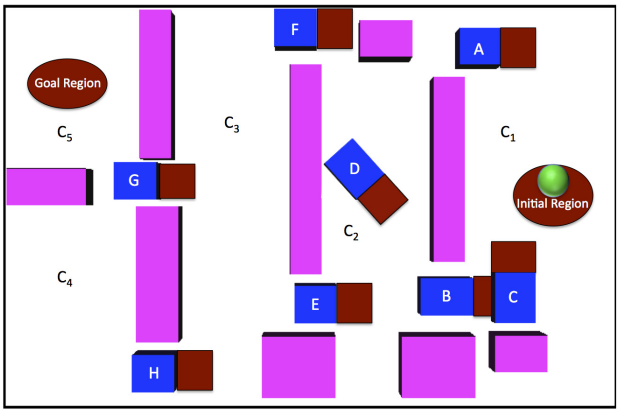

Fig. 1: Manipulation problem: the robot (green sphere) has to move from the initial to the goal region among fixed and manipulatable obstacles (labeled according to their weight in a decreasing order). Workspace regions are labeled with the names of the associated configuration space being disconnected regions.

B. Problem Formulation

Let a manipulation planning problem $\mathcal{T}$ be defined as the tuple $\mathcal{T} = \langle \mathcal{I}, \mathcal{G}, \mathcal{K} \rangle$ where $\mathcal{I}$ and $\mathcal{G}$ are, respectively, the initial and goal states, and $\mathcal{K}$ is the ontological abstract knowledge about the manipulation world.

A state is the tuple $S = \langle L, \mathcal{W} \rangle$ comprising a conjunction of literals L formed based on predicates applied to arguments and that are true in the state, and the geometric information of the workspace $\mathcal{W}$ describing the location of obstacles and the configuration of the robot. A state changes when an action is applied. An action a can be defined by a tuple $a = \langle$*name, pre, effect*$^+$, *effect*$^-$, $\mathcal{Q}\rangle$ where: *name* is a symbolic name; *pre* includes a conjunction of literals which must hold for the action to be launched; *effect*$^+$ and *effect*$^-$ are, respectively, the sets of literals to be added to or deleted from L after the action is performed; and $\mathcal{Q}$ is a query to a physics-based motion planner acting on $\mathcal{W}$, that computes a path and its actual cost, and returns the new state of the workspace. However, $\mathcal{Q}$ is not called for all actions but only for some of them.

For a given action a, the literals defining the successor state are computed as:

$$Succ(S.L, a) = S.L \cup \mathit{effect}^+(a) \backslash \mathit{effect}^-(a)$$

The geometric information $\mathcal{W}$ is updated with $\mathcal{Q}$, or left unchanged if $\mathcal{Q}$ is not called.

The following literals are used to define states, being evaluated based on a reasoning process:

- *HasAcc(FromRgn, ToRgn)*: Captures the result of geometric reasoning and evaluates to true if a trajectory may exist for the robot to move between regions *FromRgn* and *ToRgn*.
- *At(Robot, Rgn)*: Informs whether the *Robot* has reached region *Rgn*.

- *IsCritical(MObs)*: Holds if *MObs* is a manipulatable obstacle whose removal makes two disjoint configuration space regions to be connected.
- *Located(MObs, Position)*: Holds if *MObs* is located at *Position* after displacement.
- *IsManipulatable(MObs)*: Informs whether *MObs* is a manipulatable obstacle.

The three type of actions considered and their preconditions and positive and negative effects are:

- *Transit(Rob, FromRgn, ToRgn):*
  Precondition: *At(Rob, FromRgn), HasAcc(FromRgn, ToRgn)*
  Add: *At(Rob, ToRgn)*
  Delete: *At(Rob, FromRgn)*

- *Push/Pull(Rob, MObs, FromPos, ToPos, MRgn, ToRgn):*
  Precondition: *At(Rob, MRgn), IsManipulatable(MObs), IsCritical(MObs), Located(MObs, FromPos)*
  Add: *Located(MObs, ToPos), HasAcc(MRgn, ToRgn)*
  Delete: *Located(MObs, FromPos), IsCritical(MObs)*

Note that the push/pull actions are applied to have access to a given single region *ToRgn*, therefore several push/pull actions of the same object will be considered if we are interested in being able to have access to different regions.

C. Solution Overview

In order to solve the aforementioned problem, physics-based heuristics manipulation planning using knowledge is proposed as illustrated in Fig. 2, based on the Fast Forward task planning procedure (FF). The FF planning procedure does an heuristic search in the state space, where the heuristic used to estimate the cost to reach the goal from the state being evaluated is done using the *Relaxed Planning Graph* (RPG), which is a version of the Planning Graph that does not consider the negative effects. The selection of the next state in the exploration is done with *Enforced Hill Climbing* (EHC). The variant of FF that we propose uses, on the one hand, an offline reasoning process to incorporate knowledge to the manipulation problem, related to the workspace and to the manipulation of objects. Knowledge is represented by ontologies using the Web Ontology Language (OWL). On the other hand, the proposal considers an online reasoning process and the use of a physics-based motion planner to evaluate the heuristics that guides the search in the state-space. The relevant issues of the three main parts are:

- The *offline reasoning process* is responsible of using the knowledge to set the literals defining the initial state, as well as to build a graph, called $\mathcal{R}$, to define the connectivity of the workspace and that will be used for the online reasoning process.
- The *Relaxed Planning Graph* module contains the procedure to build the RPG to compute the heuristic value. The construction of the RPG uses costs of actions that take physical properties of the objects into account, and the actions appearing in the obtained relaxed planning plan are evaluated with a physics-based motion planner in order to check the feasibility and to compute the heuristic value in terms of power consumption of actions. Besides, the RPG construction is aided by the online reasoning-process that evaluates the effects of the push and pull actions with respect to the accessibility of the robot to different regions after their execution.
- The *State Space Heuristic Search* module gets $\mathcal{T}$ as input and returns either a solution plan of manipulation actions or reports failure. It keeps iteratively exploring the states using the EHC strategy and calling, for each state being explored, the RPG module to estimate the physics-based heuristic cost to reach the goal, as well as the feasible helpful actions to follow.

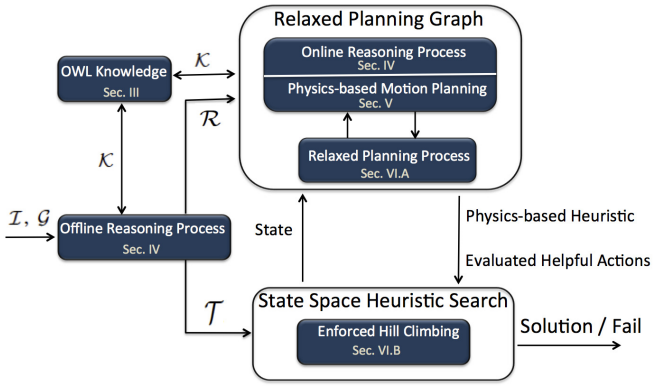

Fig. 2: Variant of the FF planning schema with indications of the sections where the different proposed components are explained.

After this overview, the paper is organized as follows. The knowledge representation is explained in Sec. III, the offline and online reasoning processes in Sec. IV and the physics-based motion planner in Sec. V. Then, Sec. VI details the algorithms of all the planning process, Sec. VII shows implementation issues and simulation results and finally Sec. VIII sketches the conclusions.

III. Manipulation Semantic Knowledge

Integrating knowledge within manipulation planning hands over to the robot awareness that may allow it to accomplish human-like tasks in which different sort of objects have to be manipulated in a skillful manner. Ontology models can be used to collect and classify knowledge within a specific domain, and to provide access to this once required. Ontologies can be developed by the *Protégé* editor [22] and encoded with the Ontology Web Language (OWL) [23] with the purpose of sharing knowledge upon multiple systems through the world wide web. In the main, OWL categorizes knowledge over classes, where collection of objects are stored, and individuals, which entail classes elements and that can correspond one another by means of properties.

To represent sufficient manipulation knowledge for the robot, a taxonomy involving two main classes, *ManipulationWorld* and *ManipulationPlanning*, have been defined in the form of OWL[1]. The *ManipulationWorld* class comprises knowledge regarding the manipulation world and is subdivided into the following classes:

- *ObjectsClassification*: Class that collects information about the objects type (e.g. fixed or manipulatable) and its physical properties such as mass, friction coefficient, etc.
- *RobotProperties*: Class that describes the robot properties and capabilities in terms of bounds of forces and torques, and of velocities and accelerations.
- *Regions*: Class that represents different types of regions including manipulatable regions of objects, initial region and goal region.
- *Perception*: Class that depicts the semantic map information including the transformation matrices that locate the objects.

The *ManipulationPlanning* class comprises knowledge related to task planning requirements and is subdivided into the following classes:

- *ProblemQueryConditions*: Class that includes the conditions about the initial and goal states of the manipulation problem that can be set by a user. According to the example, the initial state contains the initial region along initial positions of objects and the goal state involves the goal region of the robot.
- *Predicates*: Class that expresses a number of predicates (e.g. *At, HasAccess*) with associated arguments that are used inside task planning.
- *ActionProperties*: Class used to define actions and bind them with their requirements *Pre(a)*, *effect*$^+$*(a)* and *effect*$^-$*(a)*. The set of actions will be called $\mathcal{K}.\mathcal{A}$.

IV. Reasoning on Symbolic Literals using Knowledge and Geometric Information

The reasoning process consists of two steps: offline and online. The offline is used to evaluate action conditions and assert valid literals to the initial state $\mathcal{I}$ in order to avoid applying inessential actions, like the manipulation of objects that do not result in an increase of the connectivity of the configuration space, or infeasible actions, like trying to manipulate objects beyond the robot capabilities. The online aims to determine the truth of literals after actions are applied.

In order to address the offline step, a graph representing the disconnected regions of the free configuration space is first computed (this is done with the approach proposed in [16] that has been adapted to use the OWL knowledge). The nodes of this graph are regions, and any two nodes are connected by an edge if there exist an *MObs* whose motion may lead the two disconnected regions to become a single connected one. The edge is then labeled with this manipulatable obstacle, that is called a *critical* object. Also, a small circle at the end of the edge illustrates from where the object can be manipulated (this depends on which region the *MRgn* of the object lies, which can be both if the object has more than one *MRgn*, only one, or even none of them like object G in Fig. 1). Fig. 3 illustrates the graph corresponding to the example of Fig. 1, where the initial and goal nodes are highlighted as $I(C_1)$ and $G(C_5)$.

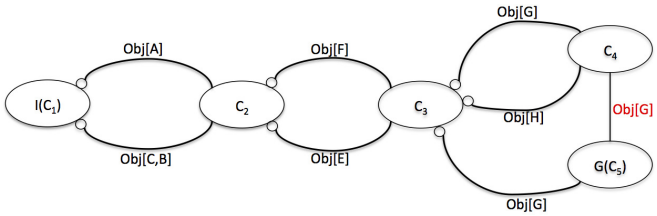

Fig. 3: Disjoint components of C_{free}

The offline reasoning process also determines:

- Whether the *MRgn* of critical objects are occupied by other *MObs*. If this is the case, these *MObs* are also labeled as critical, like object C.
- Whether the robot has access, from the initial region, to the *MRgn* of the critical objects located at $I(C_i)$ (this is done by setting *HasAcc* literal for the initial state). For instance, according to Fig. 1, literals *HasAcc(initRgn, MRgnObjectA)* and *HasAcc(initRgn, MRgnObjectC)* are initially valid.
- Whether the robot capability allows it to manipulate critical objects, taking into account their physical properties.

An online reasoning process is carried out during the construction of the RPG in order to consider only those push/pull actions according to the following criteria:

- If the obstacle being pushed/pulled is occupying the *MRgn* of another obstacle, then a push/pull action is considered with *ToRgn* set to that occupied *MRgn*. For instance, the pull of object C with *ToRgn* set to the *MRgn* of object B.
- Otherwise, if the obstacle being pushed/pulled is labeling an edge of $\mathcal{R}$ between nodes C_i and C_j, and it is manipulated from C_i, then for each critical object in C_j a push/pull action is considered with *ToRgn* set to the corresponding *MRgn* of the critical object. For instance, the push of object F with *ToRgn* set to the *MRgn* of object G, and the push of object F with *ToRgn* set to the *MRgn* of object H.

Those push/pull actions satisfying this criteria and all corresponding transit actions will be called *applicable* actions.

V. Physics-based Motion Planning

Motion planning is generally devoted to find collision-free trajectories from the start towards the goal state in the configuration space. Physics-based motion planning, however, integrates a physics-based engine that allows considering the interaction between rigid bodies and, therefore, the consider-

[1]OWL files are accessible at: https://sir.upc.edu/projects/ontologies/.

ation of trajectories that include the purposeful manipulation of objects with actions such as push or pull.

A physics-based motion planner is used to evaluate the actions appearing in the plan extracted from the RPG when computing the heuristic value. For the transit action, it evaluates whether a dynamically feasible path exists or not. For the push/pull actions, it is used to evaluate the truth of the *HasAcc* effect. This is done by iteratively pushing/pulling the obstacle a given small amount and calling the motion planner for the existence of a path from the *MRgn* to the *ToRgn*. If the effects of actions are not met according to the result of motion planning, the related action is pruned from the task planning phase, as discussed in the next section. For instance, in the explained example, there is not any valid displacement for the pull action applied to object F that can satisfy the existence of a collision-free trajectory from the *MRgn* of this object to the *MRgn* of object G, neither for the case of the pull action in order to have access to the *MRgn* of object H.

When the actions of the RPG plan are found to be feasible, then the heuristic value will be set accordingly to the actual cost of these actions. The actual cost can be computed by the physics-based motion planners as indicated by [20], however in this paper a unified cost is proposed that determines the cost of both transit and push/pull actions with respect to power consumed:

$$\mathcal{C} = \sum_{i}^{n} \frac{\mathbf{f}_i \cdot \mathbf{d}_i}{\Delta t_i}, \tag{1}$$

where $\mathbf{f}_i$ represents the control forces applied to the robot, $\mathbf{d}_i$ becomes the resultant displacement covered by the robot, and Δt_i is the ith time interval.

Then, the heuristic value of a RPG plan with n actions will be:

$$h = \sum_{i}^{n} \mathcal{C}_i \tag{2}$$

VI. Task Planning with Physics-based Heuristics

To find the manipulation plan, the modified version of the Fast-Forward (FF) planner interwoven with the reasoning process and motion planning is proposed. It consists of two main phases: constructing primarily the Relaxed Planning Graph, taking into account the feasibility and actual cost of actions to find the heuristic value, and then using the heuristic search among state space applying Enforced Hill Climbing (EHC) guided by heuristic value.

A. Relaxed Planning Graph

1) Computing the Relaxed Planning Graph: A Planning Graph is comprised of a sequence of state-levels representing a set of literals and action-levels containing a set of actions. It also considers mutual exclusion relations among each level indicating how the combination of literals can be true at each state-level. The construction phase is launched from a state-level including the initial state of the manipulation world, so an action-level appears containing actions whose preconditions are already satisfied and they may add or remove some literals in the subsequent state. The expansion process continues till all goal conditions are met. Such type of graph is the basis of *GraphPlan* symbolic task planner. The RPG becomes the simplified version of the Planning Graph, i.e., delete list of actions are ignored, so mutual exclusion relations do not appear in the planning phase. RPG is used in the FF planner to compute an heuristic value and helpful actions to guide the search of the state space.

2) Computing the heuristic value: While the RPG is being constructed, the cost of each literal l in a state-level is set based on the cost of the action a_i that generates it plus the cost of the action's preconditions $pre(a_i)$. Since several actions can generate the same literal, the minimum cost is selected. The current proposal defines the cost according to the type of action considering physics-based information, that is:

$$cost(l) = min_{\forall a_i \mid l \in effect^+(a_i)} \{cost(a_i) + \sum cost(pre(a_i))\} \tag{3}$$

where the cost of action a_i is set to 0 for the maintenance actions (maintaining literals in the next state levels). Otherwise, it depends on the action type, with push and pull actions costs greater that the transit cost and being a function of the object mass:

$$cost(transit) = 1 \tag{4}$$

$$cost(push_i/pull_i) = \frac{m_i}{m_j} \tag{5}$$

where m_i is the mass of the critical object being pushed/pulled and m_j is the mass of the lightest critical object. The search procedure of the RPG is terminated at the first state-level where all goal conditions are met. The backward search is eventually performed from these conditions in order to find the cheapest actions sequence. Then, the actions appearing in the RPG plan are evaluated using a physics-based motion planner, and the heuristic value is their total cost, as explained in Eq. (2).

3) The Relaxed Planning Graph procedure: Algorithm 1 outlines how to compute the modified RPG and to extract the heuristic value and the helpful actions. Its features are:

- *Computing action-levels, state-levels, and the RPG plan* [lines 1-10]: The initial state-level $\mathbb{S}_0$ is formed based on the information of the state S forwarded by the EHC [line 1]. Then, action-levels $\mathbb{A}_i$ and state-level $\mathbb{S}_i$ are iteratively computed [lines 4-7] until $\mathcal{G}$ is satisfied, i.e., until a state-level satisfies all the literals defining the goal state. At each level i, for the construction of $\mathbb{A}_i$, a is added once the preconditions appear in the level $i-1$ and provided it is an applicable action [line 5], as it is defined in Sec. IV. Maintenance actions are also appended [line 6] that are those maintaining literals at each $\mathbb{S}_i$. A state-level $\mathbb{S}_i$ is formed based on the effect$^+$ added by an action-level [line 7], and the cost of literals is computed in compCost function based on equation (3) [line 8]. When $\mathcal{G}$ is satisfied, the compRPGPlan function extracts the RPG plan π'.

- *Extracting the heuristic value* [lines 11-19]: This process is responsible of evaluating the actions found in π'. A call to motion planning is set by $\mathcal{Q}$ that evaluates action along the effects in order to compute the path and the cost $\mathcal{C}$ if actions effects are met [line 13]. If not, the corresponding a is pruned from the action space and does not apply again inside RPG planning [line 14] and the *RPG* function is restarted [line 15].
- *Computing helpful actions* [line 20]: Finally, the helpful actions $H(S)$ are retrieved in the extractHActions function (they are the actions in π' appearing in the first action level).

Algorithm 1 RPG$(S, \mathcal{G}, \mathcal{K}, \mathcal{R})$

```
1:  𝕊_0 ← S
2:  i ← 0
3:  while i <MaxLevels do
4:      i ← i + 1
5:      𝔸_i ← {a ∈ 𝒦.𝒜 | pre(a) ⊆ 𝕊_i-1 & applicable(a, ℛ)}
6:      𝔸_i ← append(maintActions(𝕊_i-1))
7:      𝕊_i ← {l | l ∈ effect+(𝔸_i)}
8:      compCost(𝕊_i)
9:      if 𝒢 ⊆ 𝕊_i then
10:         π' ← compRPGPlan()
11:         h ← 0
12:         for each a ∈ π' do
13:             if !𝒬(a, 𝒞) then
14:                 pruneActions(a)
15:                 GOTO 4
16:             else
17:                 h ← h + 𝒞
18:             end if
19:         end for
20:         H(S) ← extractHActions(π')
21:         return {h, H(S)}
22:     end if
23: end while
24: return NULL
```

B. State space heuristic search

The FF planner searches the state space using the RPG heuristic and employing the Enforced Hill Climbing (EHC). The procedure of finding a feasible plan is sketched in Algorithm 2. First of all, symbolic literals are evaluated based on the offline reasoning process for obtaining $\mathcal{R}$ and the initial state of the manipulation world S_0 for EHC as is done in offlineProcess [line 2]. The process of EHC [lines 3-9] is the one used in the standard FF. From a given state, the *RPG* function is called in order to compute the $H(S_i)$ and h [line 4]. selectHAction is then responsible to return the helpful action $H'(S_i)$ reducing the heuristic value for the current state [line 5]. The selected action is appended to the plan π [line 6] and the next state is formed by Succ [line 7] for the subsequent iteration. The iteration is carried out untill all conditions of $\mathcal{G}$ are satisfied and the final plan π is eventually achieved.

Algorithm 2 Feasible plan extraction

Input: $\mathcal{I}, \mathcal{G}, \mathcal{K}$
Output: The feasible plan π

```
1: π ← ∅
2: {S_0, ℛ} ← offlineProcess(ℐ, 𝒢, 𝒦)
3: while 𝒢 ⊈ S_i do
4:     h(S_i), H(S_i) ← RPG(S_i, 𝒢, 𝒦, ℛ)
5:     H'(S_i) ← selectHAction(H(S_i), h(S_i))
6:     π.append(H'(S_i))
7:     S_i+1 ← Succ(S_i, H'(S_i))
8: end while
9: return π
```

VII. Implementation and simulation results

A. Implementation

The proposed framework implementation consists of two major layers: task-level and motion-level layers. At the task-level layer, geometric reasoning and the OWL developed by the Protégé editor are applied to formulate the manipulation requirements. The task planning algorithm is implemented using the Prolog language that is able to fetch the OWL knowledge through the Knowrob software which is a powerful tool to enable the access of stored knowledge by Prolog predicates [24]. At the motion-level layer, The Kautham Project [25] is used. It is an open source motion planning tool based on C++ that enables to plan under kinodynamic and physics-based constraints. It uses Open Motion Planning Library (OMPL) [26] as its core of planning algorithms. OMPL provides a wide set of sampling based motion planners such as RRT and KPIECE [27]. It is also integrated with the Open Dynamic Engine (ODE) for the dynamic simulations and ontological knowledge as presented in [19]. Communication among these planning layers are performed using ROS [28].

B. Simulation results

The proposal has been simulated for the example in Fig. 1 and the sequence of actions for the execution of the plan is depicted in Fig. 4. KPIECE is used as kinodynamic motion planner for the physics-based planning since it has the highest success rate and computes the time-optimal solution as compared to other state-of-the-art kinodynamic planners such as RRT and EST [29]. While manipulation planning is performed, some potential actions are found in RPG plans, being their feasibility under investigation by the online reasoning process. It has been observed, for instance, that when RPG planning process is taking place, it confronts with several potential parallel actions like *Push/PullE* and *Push/PullF*, and the one of least cost is selected, in this case actions involving object F which is lighter than object E. Online reasoning process is then responsible to evaluate RPG plans and reject those ones containing infeasible actions (such as *PushC, PullF,* and *PushG*) because their effects cannot be satisfied. The simulation was run on an Intel Core i7-4500U 1.80GHz CPU with 16 GB memory and average planning time to compute the final manipulation plan was 89 s including several calls to the motion planner.

Fig. 4: Snapshots of the task execution that involves the following actions: 1) transit to C, 2) pull C, 3) transit to B, 4) pull B, 5) transit to F, 6) push F, 7) transit to G, 8) pull G, 9) transit to Goal (The solution can be visualized in https://sir.upc.edu/projects/kautham/videos/manip.mp4).

VIII. Conclusions

A framework to efficiently interleave task planning with physics-based motion planning has been presented based on a version of the FF planner. Different types of reasoning processes (online and offline) integrated with manipulation knowledge, the geometry of the workspace, and motion planning has been offered for evaluating actions in order to guide the task planning phase, allowing to find low-cost feasible plans. The present framework has been implemented and simulated with a manipulation problem including a mobile manipulator and different type of fixed as well as manipulatable obstacles with various features, validating the proposal. Future work is centered in enhancing the action space with pick and place actions according to the robot knowledge and to incorporate uncertainty inside the task-level.

References

[1] C. Dornhege, P. Eyerich, T. Keller, S. Trüg, M. Brenner, and B. Nebel, "Semantic attachments for domain-independent planning systems," in *Towards Service Robots for Everyday Environments*. Springer, 2012, pp. 99–115.

[2] E. Erdem, K. Haspalamutgil, C. Palaz, V. Patoglu, and T. Uras, "Combining high-level causal reasoning with low-level geometric reasoning and motion planning for robotic manipulation," in *Robotics and Automation (ICRA), 2011 IEEE International Conference on*. IEEE, 2011, pp. 4575–4581.

[3] S. Srivastava, E. Fang, L. Riano, R. Chitnis, S. Russell, and P. Abbeel, "Combined task and motion planning through an extensible planner-independent interface layer," in *Robotics and Automation (ICRA), 2014 IEEE International Conference on*. IEEE, 2014, pp. 639–646.

[4] E. Plaku and G. D. Hager, "Sampling-based motion and symbolic action planning with geometric and differential constraints," in *Robotics and Automation (ICRA), 2010 IEEE International Conference on*. IEEE, 2010, pp. 5002–5008.

[5] L. P. Kaelbling and T. Lozano-Pérez, "Hierarchical task and motion planning in the now," in *Robotics and Automation (ICRA), 2011 IEEE International Conference on*. IEEE, 2011, pp. 1470–1477.

[6] A. K. Pandey, J.-P. Saut, D. Sidobre, and R. Alami, "Towards planning human-robot interactive manipulation tasks: Task dependent and human oriented autonomous selection of grasp and placement," in *Biomedical Robotics and Biomechatronics (BioRob), 2012 4th IEEE RAS & EMBS International Conference on*. IEEE, 2012, pp. 1371–1376.

[7] L. de Silva, A. K. Pandey, M. Gharbi, and R. Alami, "Towards combining HTN planning and geometric task planning," *arXiv preprint arXiv:1307.1482*, 2013.

[8] M. Gharbi, R. Lallement, and R. Alami, "Combining symbolic and geometric planning to synthesize human-aware plans: toward more efficient combined search." in *Intelligent Robots and Systems (IROS), 2015 IEEE/RSJ International Conference on*. IEEE, 2015, pp. 6360–6365.

[9] F. Gravot, S. Cambon, and R. Alami, "aSyMov: a planner that deals with intricate symbolic and geometric problems," in *Robotics Research. The Eleventh International Symposium*. Springer, 2005, pp. 100–110.

[10] S. Cambon, R. Alami, and F. Gravot, "A hybrid approach to intricate motion, manipulation and task planning," *The International Journal of Robotics Research*, vol. 28, no. 1, pp. 104–126, 2009.

[11] C. R. Garrett, T. Lozano-Pérez, and L. P. Kaelbling, "Ffrob: An efficient heuristic for task and motion planning," in *Algorithmic Foundations of Robotics XI*. Springer, 2015, pp. 179–195.

[12] J. Hoffmann and B. Nebel, "The ff planning system: Fast plan generation through heuristic search," *Journal of Artificial Intelligence Research*, pp. 253–302, 2001.

[13] A. L. Blum and M. L. Furst, "Fast planning through planning graph analysis," *Artificial intelligence*, vol. 90, no. 1, pp. 281–300, 1997.

[14] M. Tenorth and M. Beetz, "A unified representation for reasoning about robot actions, processes, and their effects on objects," in *Int. Conf. on Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ*. IEEE, 2012, pp. 1351–1358.

[15] D. Di Marco, P. Levi, R. Janssen, R. van de Molengraft, and A. Perzylo, "A deliberation layer for instantiating robot execution plans from abstract task descriptions," in *Proc. of the Int. Conf. on Automated Planning and Scheduling: Workshop on Planning and Robotics (PlanRob). AAAI Press*, 2013.

[16] M. Stilman and J. J. Kuffner, "Navigation among movable obstacles: Real-time reasoning in complex environments," *International Journal of Humanoid Robotics*, vol. 2, no. 04, pp. 479–503, 2005.

[17] S. Zickler and M. Veloso, "Efficient physics-based planning: sampling search via non-deterministic tactics and skills," in *Proc. of The 8th Int. Conf. on Autonomous Agents and Multiagent Systems-Volume 1*, 2009, pp. 27–33.

[18] S. M. Lavalle and J. J. Kuffner, "Rapidly-Exploring Random Trees: Progress and Prospects," *Algorithmic and Computational Robotics: New Directions*, pp. 293–308, 2001.

[19] Muhayyudin, A. Akbari, and J. Rosell, "Ontological physics-based motion planning for manipulation," in *Proc. of IEEE Int. Conf. on Emerging Technologies and Factory Automation (ETFA)*. IEEE, 2015, pp. 1–7.

[20] A. Akbari, Muhayyudin, and J. Rosell, "Task and motion planning using physics-based reasoning," in *Proc. of the IEEE Int. Conf. on Emerging Technologies and Factory Automation*. IEEE, 2015, pp. 1–7.

[21] A. Akbari, Muhayyuddin, and J. Rosell, "Reasoning-based evaluation of manipulation actions for efficient task planning," in *Robot 2015: Second Iberian Robotics Conf.* Springer, 2016, pp. 69–80.

[22] Stanford, "Protégé," http://protege.stanford.edu/, 2007.

[23] D. McGuinness, F. Van Harmelen *et al.*, "Owl web ontology language overview," *W3C recommendation*, vol. 10, no. 10, p. 2004, 2004.

[24] M. Tenorth and M. Beetz, "Knowrob knowledge processing for autonomous personal robots," in *Int. Conf. on Intelligent Robots and Systems (IROS)*. IEEE, 2009, pp. 4261–4266.

[25] J. Rosell, A. Pérez, A. Aliakbar, Muhayyuddin, L. Palomo, and N. García, "The Kautham Project: A teaching and research tool for robot motion planning," in *Proc. of the IEEE Int. Conf. on Emerging Technologies and Factory Automation*. IEEE, 2014, pp. 1–8.

[26] I. Sucan, M. Moll, L. E. Kavraki *et al.*, "The open motion planning library," *Robotics & Automation Magazine, IEEE*, vol. 19, no. 4, pp. 72–82, 2012.

[27] I. A. Şucan and L. E. Kavraki, "Kinodynamic motion planning by interior-exterior cell exploration," in *Algorithmic Foundation of Robotics VIII*. Springer, 2010, pp. 449–464.

[28] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng, "ROS: an open-source robot operating system," in *ICRA Workshop on Open Source Software*, vol. 3, 2009, p. 5.

[29] Muhayyuddin, A. Akbari, and J. Rosell, "Physics-based motion planning: Evaluation criteria and benchmarking," in *Robot 2015: Second Iberian Robotics Conf.* Springer, 2016, pp. 43–55.